Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions

Joël Alwen
IST Austria

Jeremiah Blocki
Purdue University

Ben Harsha
Purdue University

ABSTRACT

A memory-hard function (MHF) f_n with parameter n can be computed in sequential time and space n. Simultaneously, a high *amortized parallel* area-time complexity (aAT) is incurred per evaluation. In practice, MHFs are used to limit the rate at which an adversary (using a custom computational device) can evaluate a security sensitive function that still occasionally needs to be evaluated by honest users (using an off-the-shelf general purpose device). The most prevalent examples of such sensitive functions are Key Derivation Functions and password hashing algorithms where rate limits help mitigate off-line dictionary attacks. As the honest users' inputs to these functions are often (low-entropy) passwords special attention is given to a class of side-channel resistant MHFs called iMHFs.

Essentially all iMHFs can be viewed as some mode of operation (making n calls to some round function) given by a directed acyclic graph (DAG) with very low indegree. Recently, a combinatorial property of a DAG has been identified (called "depth-robustness") which results in good provable security for an iMHF based on that DAG. Depth-robust DAGs have also proven useful in other cryptographic applications. Unfortunately, up till now, all known very depth-robust DAGs are impractically complicated and little is known about their exact (i.e. non-asymptotic) depth-robustness both in theory and in practice.

In this work we build and analyze (both formally and empirically) several exceedingly simple and efficient to navigate practical DAGs for use in iMHFs and other applications. For each DAG we:

- Prove that their depth-robustness is asymptoticly maximal.
- Prove bounds of at least 3 orders of magnitude better on their exact depth-robustness compared to known bounds for other practical iMHF.
- Implement and empirically evaluate their depth-robustness and aAT against a variety of state-of-the art (and several new) depth-reduction and low aAT attacks. We find that, against all attacks, the new DAGs perform significantly better in practice than Argon2i, the most widely deployed iMHF in practice.

Along the way we also improve the best known empirical attacks on the aAT of Argon2i by implementing and testing several heuristic versions of a (hitherto purely theoretical) depth-reduction attack. Finally, for the best performing of the new DAGs we implement an iMHF using the Argon2i round function and code base and show that on a standard off-the-shelf CPU the new iMHF can actually be evaluated slightly faster than Argon2i (despite seemingly enjoying significantly higher aAT).

CCS CONCEPTS

• **Security and privacy → Hash functions and message authentication codes**;

KEYWORDS

hash functions, key stretching, depth-robust graphs, memory hard functions

1 INTRODUCTION

A memory-hard function (MHF) is a family of functions equipped with an honest algorithm $\mathcal{N}$ for evaluating them such that $\mathcal{N}$ requires a limited amount of (sequential) computation and memory, yet no parallel amortized algorithm can significantly reduce the product of space and time required per evaluation of the MHF. Intuitively, the goal of MHFs is to limit the advantage (in terms of dollar per rate of evaluation) that an adversary equipped with a (potentially highly parallel) custom computational device such as an FPGA or an Application Specific Integrated Circuit (ASIC) has over the average honest user who only has an (essentially sequential) general purpose CPU at their disposal. In practice, MHFs are useful in applications where we would like to limit the rate at which an adversary can evaluate a particular function while simultaneously not preventing honest parties from evaluating the function at a moderate rate. An important family of such functions are found in password based cryptography. For example we may wish to limit the rate at which an adversary can evaluate a KDF or password hashing algorithm in order to increase the difficulty of the adversary performing dictionary attacks.

Several distinct hardness notions based on memory have been considered in the literature. An early example is (sequential) *space complexity* (considered for example in [32]) which lowerbounds the minimum amount of memory required to evaluate a given function on a (sequential) computational device. In the context of cryptography *memory-bound* functions were first considered in [3, 23]. There, the complexity of a function is taken to be the minimum expected number of cache misses in a computation of the function.

More recently, *memory-hard* functions (MHFs) were introduced by Percival [33] in the form of the `scrypt` algorithm. Intuitively MHFs define the complexity of a function to be the minimum product of the space and time required to compute the function (amortized across a large number of concurrent evaluation). [1] MHFs have since found growing favor in the security and cryptographic community. They have been proposed (and are being increasingly used) for password hashing for use in storing credentials in a login server and as key derivation functions in password based cryptography. (For example in the recent Password Hashing Competition [1] almost all of the entrants, including all finalists and the winner, claimed some form of memory-hardness.) MHFs are also being increasingly used as the basis for Proof-of-Work protocols underlying

[1] For a brief discussion on the difference between memory-bound and memory-hard see Appendix A.

cryptocurrencies such as Litecoin [20], Dogecoin [13], ZCash [40] and Ethereum [38].

Data (In)Dependance. MHFs can generally be divided into two categories. A data-dependent MHF (dMHF) is characterized by the property that the memory-access pattern of $\mathcal{N}$ depends on the input to the MHF. Conversely, if the property does not hold then we use the term data-independent MHF (iMHF). The latter are particularly interesting in the context of password hashing as they naturally resist a class of side-channel attacks (e.g. cache-timing attack [12]) which could potentially leak sensitive information about the inputs; namely about the password being hashed. However this resistance does come at a price; certain dMHFs such as `scrypt` have been shown to enjoy strictly greater memory-hardness [8] then can possibly be achieved for a very broad class of iMHFs [5] that include all known theoretical and practical candidate iMHF constructions.

iMHFs and Graphs. In a bit more detail, this class of iMHFs can be characterized as iMHFs consisting of some static mode of operation over (a fixed input-length) compression function. Both in theory and in practice essentially all iMHFs are designed (or can be viewed) as such. Whats more, the particular mode of operation can in turn be viewed as directed acyclic graph (DAG) which describes how the inputs and outputs of various calls to an underlying compression function are related to each other. First used in [24] this method of describing a function in terms of a DAG and compression function has witnessed various incarnations both explicit [7, 10, 25, 26, 28, 30] and implicit [4, 15, 18, 19, 22, 34, 39] to name just a few.

Put simply, the iMHF f_G given by DAG G and round function h is computed by labeling the nodes of G. The input(s) to f_G are the labels of the source nodes of G.[2] The labels of internal node v is computed by applying a fixed round function h to the labels of the parents of v.[3] The output of f_G is the label of the sink(s) of G.[4] If G has n nodes then for any given input x, by computing the labels one at a time in topological order, algorithm $\mathcal{N}$ can evaluate the graph function $f_G(x)$ in time n using space n times the label size. In practice h is chosen to be some cryptographic hash function. Thus the memory-hardness of the graph functions is usually analyzed in the random oracle (RO) model where h is modeled as an ideal compression function (i.e. fixed input length RO).

Besides clarity gained by such a modular design, the real power of describing f_G in terms of G can be seen in [10, 11] where a lowerbound on the memory-hardness of f_G (in the parallel random oracle model) is given in terms of the "amortized area-time" pebbling complexity (aAT) of G or $\mathsf{aAT}(G)$ for short. This is a complexity notion for a DAG given by measuring cost of an optimal strategy for a game played involving placing pebbles on the nodes of G according to two very simple rules.[5] Intuitively, while any DAG on n nodes gives rise to an iMHF which takes the same amount of resources for the honest party using algorithm $\mathcal{N}$, the memory-hardness of f_G grows as does $\mathsf{aAT}(G)$. This motivates the search for simple DAGs with maximal aAT over all graphs of equal size.

Depth-Robust Graphs. Recently it has been shown that for a DAG G to have high aAT it is both necessary [7] and sufficient [5] for G to be very depth-robust.

An (e, d)-depth-robust directed acyclic graph (DAG) has the property that after removing any subset of up to e nodes (and adjacent edges) there remains a directed path of length d. By very depth-robust we essentially mean that the product of e and d should be large. The problem of constructing DAGs with extreme depth-robustness was first investigated by Erdös, Graham and Szemerédi in[27]. There, a graph on n nodes with indegree $O(\log(n))$ is constructed[6] such that for certain positive constants c_1 and c_2 removing any $c_1 n$ nodes leaves a path of length $c_2 n$. More recently Mahmoody, Moran and Vadhan [30] adapted the construction of [27] such that for every $\epsilon > 0$ and $n \geq n_\epsilon$ large enough [30] constructs a graph G_n on n nodes with indegree $\tilde{\Omega}(\log^2)$ such that for any $\alpha \in (0, 1)$ the graph is $(\alpha n, (\alpha - \epsilon)n)$-depth-robust.[7]

Originally, depth-robust graphs found theoretical applications in proving lowerbounds on circuit complexity and Turing machine time [31, 35–37] and, quite recently, to prove lowerbounds in the domain of proof complexity [2]. However, more recently [30] used depth-robust graph to construct several Proofs of Sequential Work; protocols allowing a prover to convince a computationally-bounded verifier that a certain amount of sequential work was done (despite the prover being capable of arbitrary parallel computation). Yet more recently, strong connections have emerged between depth-robust graphs and study of secure memory-hard functions (MHF). [5, 7] In particular, [7] showed that if G is (e, d)-depth-robust then $\mathsf{aAT}(G) > ed$. The more depth-robust G the more memory hard f_G becomes.

New Requirements and Constraints for Practice. In contrast to more theoretical applications of depth-robust graphs, both those in [30] and those in the study of memory-hard functions impose new requirements on the constructions of depth-robust (and high aAT) graphs. This is because both types of applications require honest parties to label a fixed concrete depth-robust graph G (though with different round functions and to different ends). Moreover the security and efficiency of the resulting constructions is tightly related indegree of G and to the maximal e and d for which G is (e, d)-depth-robust (or to the aAT G). In particular, the analogue of a security parameter in more conventional cryptographic schemes is the number of nodes of G. Thus, applications ask for a sequence of graphs of increasing size n such that their respective depth-robustness (or aAT) properties grow in n. Together these properties of the applications impose the following new desiderata for depth-robust graphs:

Low In-Degree: For the MHF applications the round function h is modeled as a (random) oracle. In particular this imposes the restriction that evaluating h requires having all inputs in memory simultaneously. In practice though h is implemented by a cryptographic hash function which

[2] A node is called a *source* if it has no incoming edges.
[3] A *parent* of v is a node u such that G contains the edge (u, v).
[4] A *sink* is a node with no out going edge.
[5] In some other works $\mathsf{aAT}(G)$ is lowerbounded by the closely related notion of "cumulative pebbling complexity" of the graph [7, 10].

[6] The indegree of a graph is the largest number of incoming edges for any node in the graph.
[7] To be precise, except for [27], the remaining works on depth-robustness from the 70s and 80s actually considered a vairant with edge removal instead of node removal. However, for constant indegree graphs, as used in this work, the two notions are effectively the same.

are iterative functions (e.g. using the Merkle-Damgård construction). This means that when the input x to h is long then there is really no need to store all of x at once. Given the importance of memory consumption to the security definition it is important to minimize this discrepancy between the RO model and real world computation. Thus, as the length of the (longest) input to h is dictated by the indegree of G, to build a memory-hard function we would like that the indegree of G be as small as possible (usually 2).

For the case of Proofs-of-Sequential-Work, the efficiency of each protocol in [30] degrades significantly in the indegree of G. Therefore, in this case too we would like to minimize the indegree.

Extreme Depth-Robustness & aAT: The security of cryptographic applications discussed above is tightly tied to the depth-robustness and aAT of the underlying DAG. Thus, a good start is to use a family of graphs with asymptotically maximal values in n for these measures. However, while asymptotics provide some evidence for soundness of a construction, in any given practical instance, it is the exact security for the concrete parameters being used that ultimately matters.

Therefore, going beyond asymptotic optimality we propose two further desiderata. First, the hidden constants in the asymptotics should be made explicit and upper bounded as far as possible. Second, we would like empirical evidence supporting the claim that the graph has high depth-robustness and/or large aAT. This can take the form of evaluating the success of state-of-the-art depth-reduction algorithms and of efficient pebbling strategies that aim to minimize the pebbling cost of the graph. (The latter algorithms can, in particular, give rise to evaluation strategies for evaluating the iMHF f_G on a parallel devices with low amortized space/time per instance. [5, 7])

Simple & Locally Navigable: In all cryptographic applications honest parties are required to label the nodes of G. For this to be made practical we would like to be able to provide implementors with a simple, elegant and concise algorithm for navigating the edge structure of the graph (in particular for determining the parents of a given node). All past constructions of graphs with extreme depth-robustness (and asymptotically optimal aAT) rely heavily on low degree expander graphs with extreme expansion properties. While in theory well understood, in practice these can prove to be either rather complicated with large (or at least poorly understood) constants describing their expansion properties (relative to indegree) or to be simple but exhibiting suboptimal expansion properties.

With the goal of wide spread adoption in mind it would be useful to avoid generic expanders altogether. More to the point, ideally we would like a graph equipped with a simple algorithm (ie. consisting only of a few basic arithmetic and control flow instructions) for computing the parent function in polylog time and space. More precisely given node $v \in V$ and $i \in \{1, \ldots, \mathsf{indeg}(v)\}$ the algorithm outputs the i^{th} parent node u of v in time (and space) $O(\log_2(|V|))$ with only very small constants hidden in the O notation.

1.1 Existing Graphs and Their Properties

Due to a Lemma by Valiant [37] it follows that for any graph G on $n = 2^k$ nodes with indegree δ and any t there exists a subset set $S \subseteq V$ of nodes of size $|S| \leq \delta t n/(\log(n) - t)$ such that removing S leaves no path of length greater than 2^{k-t}. Several constructions of graphs with low indegree exhibiting this asymptotically optimally depth-robustness are given in the literature [27, 31, 35, 36] but all rely on low (or constant) indegree expanders with extreme expansion properties. The graphs in [30] also have indegree $\tilde{\Omega}(\log^2(n))$ rather $\log(n)$ which would be optimal for graphs of equal size and depth-robustness.

Valiants upperbound on the depth-robust of a graph and the algorithm for pebbling non-depth-robust (i.e. depth-reducible) graphs in [5] together imply that any graph has aAT $O(n^2 \log\log n/\log(n))$. All graphs graphs with (near) optimal aAT are built from optimal depth-robust graphs [7, 10].

In contrast, for graphs with simple, locally navigable constructions which do not rely on general expanders the asymptotics fall well short of what we could hope to achieve. In particular, the aAT of graphs used in iMHFs proposed for use in practice have so far proven to be well below optimal [5–7, 9]. Of these the most prominent is Argon2i [15] which won the recent Password Hashing Competition and is rapidly establishing itself as the most widely deployed iMHF in practice. We destinguish between the latest version (1.3) Argon2iB and any previous versions Argon2iA as they rely two different families of DAGs. In particular, for the graphs underlying Argon2iB and Argon2iA the results in [7] show their aAT to be $O(n^{1.8})$ and $O(n^{1.708})$ respectively. Graphs for other iMHFs such as both Catena functions and the Balloon Hashing functions have aAT $O(n^{1.67})$ or even $o(n^{1.625})$. Similar results [9] hold for the graphs used in Pomelo [39], Lyra2 [4], Rigv2 [19], Gambit [34] and TwoCats [22].

In terms of constants, to the best of our knowledge, no effort has been made for depth-robust graphs nor for high aAT graphs to optimize the construction (or their analysis) to achieve good constants. In fact no such analysis has been done prior to this work. In Appendix D we show that for the depth-robust graph of [27] the proof in that work implies that the (constant indegree version of the) graph is (e, d)-depth-robust for $ed = cn^2/\log(n)$ where $c = 4.72 \times 10^{-7}$. The graph underlying the Argon2iA is only known [5] to have aAT $n^{5/3}/(c \log^2 n)$ for $c = 9.6 \times 10^7$.

1.2 Our Results

In a nutshell, in this work we make progress towards bringing the constructive applications of depth-robust and high aAT graphs into practice. First we build and analyze a very simple and locally navigable sequences of graphs with indegree 2 and asymptotically optimal depth-robustness. We given an upper bound for the hidden constants an order of magnitude smaller then any known even for the best theoretical construction ([27]). We also give a second construction with similar properties but for high aAT. For this we upper bound the hidden constants in its aAT to be two orders of

magnitude smaller than the best known for any graph with optimal asymptotic aAT.

Finally we implement the graphs in software and run a variety of state-of-the-art depth-reduction and low aAT attacks for graph sizes used in practice. We compare the the results very favorably to those of similar experiments for the DAGs underlying Argon2iA and Argon2iB. In particular the empirical results indicate that the hidden constants for the depth-robustness and aAT of our constructions are vastly smaller than we are able to bound rigorously. In more details we do the following.

DRSample. In Section 3 we introduce algorithm DRSample for sampling the parents of nodes in a random DAG G of size n. Next we prove a probabilistic bound on the parameters of the depth-robustness of G. In particular, G is $(\Omega(n/\log n), \Omega(n))$-depth-robust except with negligible probability which is optimal. Our proof actually shows that G satisfies a stronger variant of depth-robustness called block depth-robustness [7] with block sizes of length $b = \Omega(\log n)$. Intuitively, this ensures that $G - S$ contains a long path p which maintains distance b from S meaning that for any $x \in p$ none of nodes $[x, x + b]$ are contained in S.

aATSample. In Section 4 we introduce algorithm aATSample which modifies an arbitrary locally navigable block depth-robust DAG G with a simple operation so as to obtain a new locally navigable graph G'. We show how to transform an exact (i.e. non asymptotic) lowerbound on the aAT of G into a higher exact lowerbound on aAT of G'. In particular we can use DRSample for G and, combining the results from the previous section with those in [7] relating depth-robustness to aAT, we obtain the necessary exact lowerbound on the aAT of G.

Empirical Analysis. In Section 5 we provide empirical evidence for the suitability of both families of graphs.

Implemented Attacks: For attacks we implement 6 depth-reducing algorithms; that is algorithms for finding small sets S of nodes for reducing the depth of a given graph. The first is taken from the attack in [5] while the other four are variants of Valiant's algorithm in [37]. The last one (called "Best Attack") simply takes the smallest set S found by any of the other algorithms for a given target depth.

We also implement the parallel pebbling attack of [5] which produces a low aAT pebbling of a given DAG G and depth-reducing set S. In particular the algorithm makes use Best Attack as a subroutine and searches for optimal parameters for running the [5] pebbling algorithm.

Implemented Graphs: Next we implement 8 graphs, Argon2i-A, Argon2i-B, DRSample, aATSample as well as a variant of the latter requiring less randomness to sample (a valuable commodity in some practical applications) but for which the formal analysis about the constants (and asymptotics) carry over almost unchanged. Finally, for applications where no randomness is available at all, we also implement fully deterministic "Powers-of-Two" graph. This latter graph is also *exceedingly* simple and efficient to implement both in hardware and software requiring nothing other then a single mod (potentially even a power of 2) operation and one bit-shift to compute the parent function.

Security: Our first contribution in this section is to show that, in practice, depth-reduction techniques based on Valiant's lemma actually outperform the Layered depth-reduction attacks of [5]. At least in the cases of the Argon2i graphs this is somewhat surprising as asymptotic analysis of Layered for those particular graphs indicates that it should out perform significantly better then asymptotics known to hold for Valiant's lemma (albeit on an arbitrary graph). In practice, the converse seems to be true, at least for Argon2i-A, Argon2i-B, DRSample and the Powers-of-Two graph. (We believe this to indicate that, even in theory, the behavior of Valiant's lemma merits further investigation.)

Next, we describe and analyze the results of running the Best Attack depth-reducing algorithms and against each of the graphs on sequence of interesting graph sizes for use in practice. We found that all new constructions fair significantly better than Argon2i-B. Amongst the new constructions DRSample seems to be the most depth-robust of all. For example, in order to reduce the depth of the Argon2i-B graph on $n = 2^{24}$ nodes to approximately 4×10^6 a set of size $|S| = 6.7 \times 10^3$ was found while for DRSample no set smaller than $|S| \geq 12 \times 10^5$. Recall that when G is (e, d)-depth-robust then $\mathsf{aAT}(G) > ed$. For Argon2i-B the (e, d) point with the highest such product we found was 2.5×10^{11} while for DRSample we found a point with $ed \approx 5 \times 10^{12}$.

Finally, we report and analyze the results of running the pebbling attack against each graph (for each size) on input the optimal depth-reducing set found for that choice of graph and size. Here too the new constructions faired better than either version of Argon2 (though the fully deterministic construction only marginally so). Once again DRSample and its variants proved the most resilient though aATSample was not much worse. For example, when $n = 2^{24}$ (which corresponds to 8GB of memory when using Argon2's compression function which can be computed in roughly 1.25 seconds) we see the aAT of the attack on Argon2i-B is roughly 11.5 times better than the honest party while against DRSample the improvement is only 3.4 fold.

Improved Attacks: Along the way we also improve the best known empirical results for low aAT and depth-reduction attacks on both Argon2i-A and Argon2i-B compared with the state-of-the-art in [6]. For example, when evaluating Argon2i-B with one pass over 8GB of memory our new low aAT attack is now almost 11.5 times more efficient than the honest algorithm (compared to 9.3 times more efficient in [6]). For a similar evaluation of Argon2i-A we improve from 14.2 to 19 times the honest algorithms efficiency. This may be somewhat unexpected in light of the fact that, compared to Valiants lemma, the best (known) lowerbounds for the attacks on those graphs are actually better for the Layered depth-reduction algorithm used in [6]. Never-the-less, in practice our experiments indicate that Valiant's approach seems to work better.

Timing: Finally we also report on experiments measuring the time required by the honest evaluation algorithm for evaluating an iMHF obtained by using DRSample with Argon2iB's compression function on an off-the-shelf general purpose CPU. We show that for the same number of calls to the underlying round function the new iMHF can be evaluated as slightly faster than Argon2iB.

1.3 Discussion

To be clear, although we believe these results represent significant improvements in terms of practical applicability compared to past graph constructions, the constants for the depth-robustness and aAT which we are able to prove still leave something to be desired for graph of the size we would like to use in practice (e.g. iMHF $n = 2^{22}$ is a reasonable value). However, the empirical results strongly indicate that an iMHF using our new constructions are both as (or even more) efficient for the honest user while simultaneously resulting in significantly greater memory-hardness when compared to state-of-the-art in practical iMHFs. With this in mind we view the theoretical techniques introduced in this work for bounding the constants of the new constructions as a strong starting point for the further investigation into tightening the bounds. We conjecture that the graphs presented here do in fact achieve constants of practical interest as evidenced by the failure of otherwise powerful attacks.

2 PRELIMINARIES

We begin with some notation and definitions central to this work. We denote the set of natural numbers by $\mathbb{N} = \{1, 2, \ldots\}$. For $a \leq b$ both in $\mathbb{N}$ we denote the set $\{a, a+1, \ldots, b\}$ by $[a, b]$. In particular $[1, n]$ is denoted simply by $[n]$. We denote the set of w-bit strings as $\mathbb{B}_w = \{0,1\}^w$ and the set of all bitstrings by $\mathbb{B} = \cup_{w \in [n]} \mathbb{B}_w$. We use $\log x = \log_2 x$ to denote the base 2 logarithm.

For a directed acyclic graph (DAG) $G = (V, E)$ the *indegree* of a node $v \in V$ is the number of incoming edges. That is $\mathsf{indeg}(v) := |\{(u, v) \in E\}|$. Conversely, the *outdegree* of v is the number of outgoing edges $\mathsf{outdeg}(v) := |\{(v, u) \in E\}|$. More generally, the indegree of G is $\max\{\mathsf{indeg}(v) : v \in V\}$. A node with $\mathsf{indeg}(v) = 0$ is called a *source* and a node with $\mathsf{outdeg}(v) = 0$ is called a *sink*. We write $\mathbb{G}_n$ for the set of all DAGs on n nodes and $\mathbb{G}_{n,\delta} \subseteq \mathbb{G}_n$ for the set of DAGs with indegree δ. The *length* of a (directed) path $p = (v_1, v_2, \ldots, v_z)$ in G is the number of nodes it traverses $\mathsf{length}(p) := z$. The depth of G is the length of the longest directed path in G. The *parents* of a node v is the set $\mathsf{parents}(v) := \{u \in V : (u, v) \in E\}$ of nodes with an outgoing edge leading v and similarly the *children* of v are the nodes $\mathsf{children}(v) := \{u \in V : (v, u) \in E\}$ with an incoming edge from v. Continuing the analogy, the *ancestors* of v are all nodes with a directed path from u to v. That is $\mathsf{ancestors}_G(v) := \{u \in V : (u, \ldots, v) \text{ a path in } G\}$. When the graph G is clear from context we omit the subscript. Finally, for the sake of brevity, in this work, when we say that a set of nodes S is being removed from a graph G we implicitly also mean that incident edges to those nodes are removed. We denote the resulting graph by $G - S$.

The following (parametrized) combinatoric property of the edge structure of a DAG is central to this work. For large values of the parameters it captures the intuition that the graph remains deep even even when large arbitrary subsets of the nodes are removed.

Definition 2.1 (Block Depth-Robustness). For $n \in \mathbb{N}$ let $G = (V, E)$ be a DAG with an implicit number of its nodes $V = [n]$. Given $S \subseteq V$ let $N(S, b) = \bigcup_{v \in S}[v - b + 1, v]$. We say that a DAG G is (e, d, b)-block depth-robust if

$$\forall S \subseteq V \; |S| \leq e \Rightarrow \mathsf{depth}(G - N(S, b)) \geq d.$$

When $b =$ we simply say that G is is (e, d)-depth-robust.

We will utilize the following lemma in our security analysis. Lemma 2.2 is a slight generalization of a result of Valiant [37], and we refer to attacks based on this lemma as Valiant's Lemma Attacks.

Lemma 2.2. *Let base $b \in \mathbb{N}_{\geq 2}$ be given and let $G = (V = [n], E)$ be* any *n node DAG with $\mathsf{depth}(G) \leq b^d$ and maximum indegree $\mathsf{indeg}(G) \leq \delta$ then there is an efficient algorithm to compute subsets $S_1, \ldots, S_d \subseteq V$ with the following properties:*

(1) *For all $T \subseteq [d]$ we have $\mathsf{depth}\left(G - \bigcup_{i \in T} S_i\right) \leq b^{d - |T|}$.*
(2) *$\sum_{i=1}^{d} |V_i| \leq \delta n$.*

2.1 Graph Pebbling

The results in [10] showed how to construct provably secure MHF from any graph with high aAT. With that in mind we now formally define this complexity notion laying the groundwork for the analysis of our second construction. aAT is defined in via the parallel black pebbling game, a natural generalization to a parallel setting of the original (sequential) black pebbling game [21, 29]. The game is played in a sequence of moves consisting of placing and removing pebbles on the nodes of a given DAG according to certain (very simple) rules until all target nodes have been pebbled.

The complexity of such an execution is the sum of two values. The first summand is the sum of the number of pebbles on the graph across all steps which is called the "cumulative pebbling complexity" (CPC) of the execution. The second summand is the sum of number of times a pebble is placed on the graph. Intuitively, CPC the amortized cost of storage (i.e. storing a label for one time step) while the second term captures the amortized cost of computation. Before being added to CPC, the second summand is multiplied by the *core-memory ratio R*. This ratio is a parameter of the complexity notion aAT denoting the ratio between the cost of computation vs. storage. More precicely R is the on-chip area of a circuit computing the compression function divided by the on-chip area required to store one label. In the case of Argon2's compression function and labels the authors proposed $R = 3000$ as a realistic setting for that parameter [16] (which is the value we used in all of our experiments). For more intuition and in-depth explanation for aAT we refer the to [16] and [5] (where the notion is referred to as "energy" complexity).

We fix our notation for the parallel graph pebbling game following [5].

Definition 2.3 (Parallel/Sequential Graph Pebbling). Let $G = (V, E)$ be a DAG. A *pebbling configuration* is set $P_i \subseteq V$. Let $S, T \subseteq V$ be pebbling configurations. A *pebbling $P = (P_0, P_1, P_2, \ldots, P_t)$ with starting configuration $P_0 = S$ for target T* is a sequence of pebbling configurations such that all target nodes are pebbled:

$$\forall v \in T \; \exists z \leq t \; : \; v \in P_z.$$

The pebbling P is called *legal* if pebbles are only places on nodes whose parents are already pebbled:

$$\forall i \in [t] \; : \; v \in (P_i \setminus P_{i-1}) \; \Rightarrow \; \mathsf{parents}(v) \subseteq P_{i-1}.$$

The pebbling P is called *complete* if $S = \emptyset$ and T is the set of sinks of G. For a sequential pebbling we add the constraint that $|P_i \setminus P_{i-1}| \leq 1$, while no such constraint applies for a parallel pebbling.

Let Π be the set of all legal and complete parallel pebblings of G. Then for (implicit) core-memory ratio $R > 0$ the *cumulative pebbling complexity* (CPC) and the *amortized area-time complexity* (aAT) of a pebbling P and graph G are defined to be:

$$\mathsf{cpc}(P) := \sum_{i \le t} |P_i| \qquad \mathsf{cpc}(G) := \min\{\mathsf{cpc}(P) \ : \ P \in \Pi\}$$

$$\mathsf{aAT}(P) := \mathsf{cpc}(P) + R * \sum_{i \in [t]} |P_i \setminus P_{i-1}|$$

$$\mathsf{aAT}(G) := \min\{\mathsf{aAT}(P) \ : \ P \in \Pi\}.$$

More generally, let Π_T denote the set of legal parallel pebblings of G with target set T and starting configuration $P_0 = \emptyset$. The *cumulative pebbling complexity* (aAT) of pebbling a graph G with target set T is defined to be:

$$\mathsf{aAT}(G, T) = \min\{\mathsf{aAT}(P) \ : \ P \in \Pi_T\}.$$

For the sake of brevity, when it is clear from the context that a pebbling is legal and complete we will refer to it as simply a *pebbling of G*.

Clearly for any pebbling P (and thus for any graph) it holds that $\mathsf{aAT}(P) \ge \mathsf{cpc}(P)$ (regardless of the core-energy ration) and so a lowerbound on CPC is also a lower bound on aAT.

We will need the following result from [7].

Theorem 2.4 (Corollary 2 in [7]). *Given a DAG $G = (V, E)$ and subsets $S, T \subset V$ such that $S \cap T = \emptyset$ let $G' = G - (V \setminus \mathsf{ancestors}_{G-S}(T))$. If G' is (e, d)-depth robust then the cost of pebbling $G - S$ with target set T is $\mathsf{aAT}(G - S, T) > ed$ for any core-energy ration $R \ge 0$.*

Finally we use the notion of *quality* from [5] to evaluate how good a given pebbling strategy P is. Intuitively quality captures the multiplicative advantage of an attacker compared to the honest (sequential) evaluation algorithm. More precisely, if $P_{\mathcal{N}}$ is the honest pebbling strategy for a DAG G then the quality of pebbling P for that DAG is given by $\mathsf{aAT}(P_{\mathcal{N}})/\mathsf{aAT}(P)$. In other words, if P has quality 10 then an attacker evaluating the iMHF f_G based on the pebbling strategy P will have 10 times less amortized area-time complexity than the honest algorithm.

All graphs considered in this work have the same type of honest pebbling strategy; namely pebble the nodes one at time in topological order never removing a pebble from the DAG. Thus in each case $\mathsf{aAT}(P_{\mathcal{N}}) = n(n+1)/2 + Rn$. In particular, following the recommendation of [16] we used $R = 3000$ in our experiments.

3 A SIMPLE VERY DEPTH-ROBUST GRAPH

In this section we give the main construction (c.f. Algorithm 1) which is a very simple and efficient algorithm for sampling a DAG from a particular distribution enjoying extreme depth-robustness with high probability. It is clear by inspection that DRSample only returns acyclic graphs of size n and indegree (at most) 2. It is also easy to see that the graphs are simple and locally navigable; that is the GetParent function, which returns the i^{th} parent of a node v, requires $O(\log(n))$ simple arithmetic operations.

We prove a bound on the depth-robustness parameters of graphs sampled by DRSample in terms of n. At the highest level, the proof follows that in [27]. However we depart in several ways. First, we consider a different graph than [27] so, naturally, any statements that depend directly on the edge structure of the graph need to be reproven. In particular, the key lemma about the expansion properties of the graph needs a new approach (c.f. Lemma 3.3). Second, as our graphs are sampled randomly, we make probabilistic statements rather than absolute ones. Consequently, our proof technique now requires some probability theoretic techniques on top of the original combinatoric approach of [27]. Third, we have attempted to optimize the constant factors in the proof to the extent possible even if it makes the proof slightly more complex. By contrast, [27] seem to focus on obtaining a simple proof even if this simplicity comes at the cost of worse constant factors. We begin with a high level outline of the proof followed by a detailed exposition. Thus we use a new techinque to analyze even the purely combinatoric Lemma 3.2.

Algorithm 1: An algorithm for sampling depth-robust graphs.

Function DRSample($n \in \mathbb{N}_{\ge 2}$):
- $V := [v]$
- $E := \{(1, 2)\}$
- **for** $v \in [3, n]$ *and* $i \in [2]$ **do** `// Populate edges`
  - $E := E \cup \{(v, \mathsf{GetParent}(v, i))\}$ `// Get` i^{th} `parent`
- **end**
- **return** $G := (V, E)$.

Function GetParent(v,i):
- **if** $i = 1$ **then**
  - $u := i - 1$
- **else**
  - $g' \leftarrow [1, \lfloor \log_2(v) \rfloor + 1]$ `// Get random range size.`
  - $g := \min(v, 2^{g'})$ `// Don't make edges too long.`
  - $r \leftarrow [\max(g/2, 2), g]$ `// Get random edge length.`
- **end**
- **return** $v - r$

Proof Outline. The proof considers a graph G sampled by DRSample. First, we remove an arbitrary set of nodes S of size $O(n/\log(n))$ (and incident edges) from G. Next, for the remaining nodes in G we define a notion of a "good" node. Intuitively, these are nodes such that not too many of their neighbors were removed. The proof concludes by showing that $\Omega(n)$ of the remaining nodes must be good and that, with high probability, there remains a path p running through all good nodes. In particular after removing S graph G still has depth $\Theta(n)$.

To show that p likely exists we use the following term. For a pair of nodes v and u are "reachable" if there remains a (directed) path connecting u and v (either from u to v or vice versa). It is shown that for any good node, with high probability a large fraction the remaining nodes are reachable. Thus we can then show that any pair of good nodes are reachable. In particular we show that, with high probability, there is at least one node between the two good nodes that is reachable by both. Thus we can now construct p by connecting all the good nodes.

The details follow. We begin by stating the claim formally.

Theorem 3.1. *For $n \in \mathbb{N}$ let $G \leftarrow \mathsf{DRSample}(n)$. Then $\Pr[G$ is (e, d, b)–block depth-robust$] \ge 1 - \mathbf{negl}(n)$ where*

$$e \ge 2.43 \times 10^{-4} n/\log n = \Omega\left(\frac{n}{\log(n)}\right), \ d \ge 0.03n = \Omega(n),$$

$$b \geq 160 \log n = \Omega(\log n).$$

In particular,

$$\Pr\left[\mathsf{aAT}(G) > 7.3 \times 10^{-6} n^2/\log(n)\right] \geq 1 - \mathbf{negl}(n)\ .$$

Remark: For $G \leftarrow \mathsf{DRSample}(n)$ we have $\mathsf{aAT}(G) \geq ed \geq 7.3 \times 10^{-6} \frac{n^2}{\log n}$ by [7]. While the constant 7.3×10^{-6} is admittedly lower than one would desire in practice we point out that this is only a lower bound. In Section 5 we empirically demonstrate that $\mathsf{DRSample}(n)$ resists depth-reducing attacks, and appears to resist known attacks better than all other known iMHF candidates. Improving the constants from the lower bounds is indeed an important theoretical challenge. For comparison we note that the constant terms in all known theoretical lower bounds on $\mathsf{aAT}(G)$ for other iMHF constructions are also quite small. For example, Alwen et al. [7] were able to show that Argon2i-A and Argon2i-B have $\mathsf{aAT}(G) = \tilde{\Omega}\left(n^{5/3}\right)$. If we include the hidden constants then the bound becomes $\mathsf{aAT}(G) \geq cn^{5/3}/\log n$ with $c = 1.04 \times 10^{-8}$ which is two orders of magnitude smaller than the constant we are able to prove for DRSample.

The Meta-Graph. Before we analyze the block depth-robustness of G we first introduce the notion of a meta-graph [7]. As Claim 1 says G will be block depth-robust if and only if G_m is depth-robust.

Fix an arbitrary integer $m \in [n]$ set $n' = \lfloor n/m \rfloor$. Given a DAG G we will define a DAG G_m, called the *meta-graph* of G. For this we use the following sets. For all $i \in [n']$ let $M_i = [(i-1)m+1, im] \subseteq V$. Moreover we denote the first and last thirds respectively of M_i with

$$M_i^F = [(i-1)m + 1, (i-1)m + \left\lfloor m\left(\frac{1-\gamma}{2}\right)\right\rfloor] \subseteq M_i\ ,$$

and

$$M_i^L = [(i-1)m + \left\lceil \frac{1+\gamma}{2} \right\rceil + 1, im] \subseteq M_i\ .$$

We define the meta-graph $G_m = (V_m, E_m)$ as follows:

Nodes: V_m contains one node v_i per set M_i. We call v_i the *simple node* and M_i its *meta-node*.

Edges: If the end of a meta-node M_i^L is connected to the beginning M_j^F of another meta-node we connect their simple nodes.

$V_m = \{v_i : i \in [n']\}$ $E_m = \{(v_i, v_j) : E \cap (M_i^L \times M_j^F) \neq \emptyset\}$.

We remark that the parameter $0 < \gamma < \frac{1}{2}$ is a constant that we will optimize later. Claim 1, a simple extension of a result from [7], says that any path of length d in G_m corresponds to a path of length $\geq d\gamma$ in G. This suggests that we may want to select γ as large as possible. However, increasing γ reduces the probability that two meta-nodes in G_m are connected by an edge.

Claim 1. *If G_m is (e, d)-depth robust then G is $(e/2, d\gamma m, m)$-block depth robust.*

Proof. Fix any set $S \subseteq V$ of size $e/2$. We say that a node $v_i \in V_m$ in the meta-graph is unaffected by S if $M_i \cap \bigcup_{v \in S} \{v - m + 1, \ldots, v\} = \emptyset$. That is $G - \bigcup_{v \in S} \{v - m + 1, \ldots, v\}$ contains every node in the set M_i. Let $S_m \subseteq V_m$ denote the set of nodes affected by S. Formally, $S_m = \{v_i \in V_m\ :\ M_i \cap \bigcup_{v \in S} \{v - m + 1, \ldots, v\} \neq \emptyset\}$.

We now claim that $|S_m| \leq e$. To see this we observe that the set

$$\bigcup_{v \in S} \{v - m + 1, \ldots, v\}$$

can intersect at most e meta-nodes because for each $v \in S$ the set $\{v - m + 1, \ldots, v\}$ intersects at most two meta-nodes. Thus, S affects at most e nodes in G_m.

Since G_m is (e, d)-depth robust there remains a path ϕ' of length d in $G_m - S_m$. To complete the proof we observe that ϕ' corresponds to a path ϕ in $G - \bigcup_{v \in S} \{v - m + 1, \ldots, v\}$ of length

$$\mathsf{length}(\phi) \geq \gamma \mathsf{length}(\phi') m \geq \gamma dm.$$

In particular, the path ϕ goes through the middle γ nodes of the d meta-nodes corresponding to ϕ'. Since each of corresponding meta-nodes in ϕ' is unaffected by S the path ϕ is still contained in $G - \bigcup_{v \in S} \{v - m + 1, \ldots, v\}$. □

Proof of Theorem 3.1.

We begin by fixing some useful notation and terminology.

$I_v(r)$ and $I_v^*(r)$: The interval of r nodes preceding (or succeeding) v in G_m. That is $I_v(r) = [v - r + 1, v] \cap V_m$ and $I_v^*(r) = [v, v + r - 1] \cap V_m$.

$\bar{G}_m = (\bar{V}_m, \bar{E}_m)$: Fix a set $S \subset V_m = [n']$ with $|S| \leq cn'$ for some constant $c > 0$ (to be determined later) and let $\bar{G}_m = (\bar{V}_m, \bar{E}_m)$ be the graph obtained by removing nodes in S (and incident edges) from G_m.

Good node: Let $c > 0$ be a constant. Intuitively a node $v \in V_m$ is called *c-good* under S if at most a c-fraction of nodes in any interval starting or ending at v are contained in the set S. More precisely $v \in V_m$ is c-good if and only if both of the following hold:

- $\forall r \in [v]\ \ |I_r(v) \cap S| \leq cr$
- $\forall r \in [m - v + 1]\ \left|I_r^*(v) \cap S\right| \leq cr$.

We say a node $v \in V_m$ is c-bad under S if v is not c-good under S. When the set S of removed nodes is clear from context we will simply write c-bad or c-good.

Reachable node: A node $u \in \bar{V}_m$ is said to be *reachable for* $v \in \bar{V}_m$ under S if there exists directed a path in $\bar{G}_m$ connecting u to v or v to u.

$R_{v,S}(r)$ and $R^*_{v,S}(r)$: The set of reachable nodes in the preceding (or succeeding) intervals of size r around node v. More precisely $R_{v,S}(r) = \{u \text{ reachable for } v : u \in I_v(r)\}$ and $R^*_{v,S}(r) = \{u \text{ reachable for } v : u \in I_v^*(r)\}$.

Local Expander: Given $c > 0$ and $r* \in \mathbb{N}_{\geq 1}$ we say that a node $v \in V_m$ is a $(c, r*)$-local expander if for all $r \geq r*$ we have (1) for all subsets $A \subseteq I_v^*(r), B \subseteq I_{v+r}^*(r)$ of size $|A|, |B| \geq cr$ there exists an edge from A to B ($E_m \cap A \times B \neq \emptyset$), and (2) for all subsets $A \subseteq I_v(r), B \subseteq I_{v-r}^*(r)$ of size $|A|, |B| \geq cr$ there exists an edge from A to B ($E_m \cap A \times B \neq \emptyset$).

Note that the notion of c-good nodes is independent of the edge structure of G_m (and thus of G). To determine whether a node v is c-good it suffices to only consider the specific set S of removed nodes. By contrast, if a node is local expander or not is independent of a particular set S of removed nodes as this property depends only on the edge structure of G_m (i.e. of G).

The following claim states that even if a linear number of nodes are removed from G there still remains a linear number of good nodes in G_m. Their proof quite closely follows an similar argument in [27] (as the statement holds independently of the edge structure of G_m).

Claim 2. *[27, Claim 2] Let $G = (V, E)$ be an arbitrary DAG, let $G_m = (V_m, E_m)$ denote its meta-graph. Let $S \subset V_m = [n']$ denote an arbitrary subset vertices and constant $c_5 > 2|S|/n$. Then at least $n' - |S|\left(\frac{2}{c_5}\right)$ nodes in G_m are c_5-good. In particular, if $|S| \leq cn'$ then at least $n'\left(\frac{c_5-2c}{c_5}\right)$ nodes are c_5-good.*

Lemma 3.2 states that if $u < v$ are both c_5-good with respect to S and both (r^*, c_4)-local expanders then v is reachable from u in $G_m - S$. Lemma 3.2 improves upon a result of [27]. In particular, we achieve laxer constraints on the constants (e.g., $3c_4 + c_5 \leq 1$ vs. $3c_4 + 4c_5 < 1$) at the cost of a slightly longer proof.

Lemma 3.2. *Let $c_4, c_5 > 0, r^* \in \mathbb{N}_{\leq(1/c_5)}$ be given such that $3c_4 + c_5 \leq 1$. Fix a set $S \subseteq V_m$ and let $u, v \in V_m$ be given such that (1) u, v are both c_5-good, and (2) u, v are both (r^*, c_4)-local expanders. Then u is reachable for v under S.*

Proof. Let $S \subseteq V$ be given such that v is c_5-good under S. We prove by induction on $i \geq 0$ that for $r = 2^i r^* \leq n - v$ we have

$$\left|R_{v,S}(r)\right| \geq |I_v(r)|\,(1 - c_4) - |I_v(r) \cap S| \tag{1}$$

and if $r \leq v$ we have

$$\left|R^*_{v,S}(r)\right| \geq \left|I^*_v(r)\right|(1 - c_4 - c_5)\ . \tag{2}$$

First, observe that whenever $r \leq r^* \leq \frac{1}{c_5}$ we have $\left|R_{v,S}(r)\right| = |I_v(r)|$ and $\left|R^*_{v,S}(r)\right| = \left|I^*_u(r)\right|$ since the intervals $I_v(r)$ and $I^*_u(r)$ contain no pebbles by definition of a c_5-good node. Thus, our base case holds when $i = 0$ since we have $r = 2^0 r^* = r^*$. Given that equations 1 holds for $r = 2^i \cdot r^*$ we now show that the equations also hold for $r = 2^{i+1} r^*$. By the inductive hypothesis we have

$$\begin{aligned}\left|R^*_{v,S}(2^i r^*)\right| &\geq |I_v(r)|\,(1 - c_4) - |I_v(r) \cap S| \\ &\geq |I_v(r)|\,(1 - c_4 - c_5) \\ &\geq c_4\,|I_v(r)|\ ,\end{aligned}$$

where the last inequality follows because $1 \geq 2c_4 + c_5$. We can now invoke c_5-goodness of v along with the (c_4, r^*)-local expansion of v to argue that

$$\begin{aligned}\left|R^*_{v,S}(2^{i+1} r^*)\right| &\geq \left|R^*_{v,S}\left(2^i r^*\right)\right| + (1 - c_4)\left|I^*_{v+2^i r^*}\left(2^i r^*\right)\right| \\ &\quad - \left|I_{v+2^i r^*}\left(2^i r^*\right) \cap S\right| \\ &\geq (1 - c_4)\left|I^*_v\left(2^{i+1} r^*\right)\right| - \left|I_v\left(2^{i+1} r^*\right) \cap S\right|\ ,\end{aligned}$$

where the final step follows from the inductive hypothesis. A similar argument holds for 1 and both equations also holds for node u. WLOG assume that $u < v$ and that $v - u > r^*$ (otherwise the interval between u and v contains no pebbles!) and observe that for an *arbitrary* r we have $\left|R_{v,S}(r)\right| \geq \frac{1-c_4-c_5}{2}\,|I_v(r)|$ and $\left|R^*_{u,S}(r)\right| \geq \frac{1-c_4-c_5}{2}\left|I^*_v(r)\right| \geq c_4\left|I^*_v\right|$. Let $r = \left\lfloor \frac{v-u}{2} \right\rfloor$. If $\frac{v-u}{2}$ is an integer then the argument follows immediately since there must be an edge from $R^*_{u,S}(r)$ to $R_{v,S}(r)$. If v can be reached from node $u + r + 1$ and node $u + r + 1$ can be reached from node u then we are done. Otherwise we have two cases :

Case 1: Node $u + r + 1$ is not reachable from node u. In this case we note that $\left|R^*_{u,S}(r)\right| \geq c_4\left|I^*_v(r+1)\right|$ and that $\left|R_{v,S}(r+1)\right| \geq c_4\left|I^*_v(r+1)\right|$. Since, $R^*_{u,S}(r) \subseteq I_{v+r+1}(r+1)$ and $R_{v,S}(r+1) \subseteq I_v(r+1)$ and v is a (r^*, c_4)-local expanders we must have an edge from $R^*_{u,S}(r)$ to $R_{v,S}(r+1)$ which implies that there is a path from u to v.

Case 2: Node v is not reachable from node $u + r + 1$. In this case we note that $\left|R^*_{v,S}(r)\right| \geq c_4\left|I^*_v(r+1)\right|$ and that $\left|R^*_{u,S}(r+1)\right| \geq c_4\left|I^*_v(r+1)\right|$. Since, $R^*_{u,S}(r+1) \subseteq I^*_u(r+1)$ and $R_{v,S}(r) \subseteq I_{u+r+1}(r+1)$ and u is a (r^*, c_4)-local expanders we must have an edge from $R^*_{u,S}(r+1)$ to $R_{v,S}(r)$ which implies that there is a path from u to v. □

Thus far the results that we have proven have been independent of our graph DRSample. This changes with Lemma 3.3 which is central to our proof of the theorem. It states that in expectation at least $(1-c_6)n'$ meta-nodes (e.g., most meta-nodes) in G_m are c_4 local expanders. We remark that it is possible to apply concentration bounds to argue that the number of meta-nodes that are c_4 local expanders is tightly concentrated around its mean. We sketch this proof in Appendix C — see Lemma C.1.

Lemma 3.3. *Let $G \leftarrow \mathsf{DRSample}(n)$ and let $m = \tau \log n$, $r^* \geq 1$ and let $x \in [n'] = [n/m]$ be a meta-node then x is a (c_4, r^*)-local expander in the meta-graph $G_m = (V_m, E_m)$ except with probability at most $c_6 = \frac{1}{4r^*\pi^2 e^{-2} c_4(1-c_4)}\left(\frac{x r^*}{1-x}\right)$ where*

$$x = e^{\left(2c_4 \ln\left(\frac{1}{c_4}\right) + 2(1-c_4)\ln\left(\frac{1}{1-c_4}\right) - \frac{\tau(1-\gamma)^2 c_4^2}{8}\right)}.$$

Proof. Fix a node $v \in V_m$ and $r \geq r^*$. Let i be given such that $2^{i+1} \geq 2rm \geq 2^i$. Fix $X \subseteq I^*_v(r)$ and $Y \subseteq I^*_{v+r}(r)$ then we have

$$\Pr\left[X \times Y \cap E_m = \emptyset\right] \leq \left(1 - \frac{|X|(1-\gamma)}{8r\log n}\right)^{(1-\gamma)m|Y|} \leq \left(\frac{1}{e}\right)^{\frac{(1-\gamma)^2|Y||X|\tau}{8r}}$$

If we set $|X| = |Y| = c_4 r$ then we have

$$\Pr\left[X \times Y \cap E_m = \emptyset\right] \leq \left(\frac{1}{e}\right)^{\frac{(1-\gamma)^2 c_4^2 r\tau}{8}}.$$

We would like to use union bounds to show that (whp) no such sets X, Y exist. We have $\binom{r}{c_4 r}^2$ such pairs X, Y where, by Sterling's inequalities $\sqrt{2\pi}n^{n+0.5}e^{-n} \leq n! \leq en^{n+0.5}e^{-n}$, we have

$$\begin{aligned}
\binom{r}{c_4 r} &= \frac{r}{(c_4 r)!(r-c_4 r)!} \\
&\leq \frac{e r^{r+1/2}}{\sqrt{2\pi}(c_4 r)^{c_4 r+0.5}\sqrt{2\pi}(r-c_4 r)^{r-c_4 r+0.5}} \\
&= \frac{e r^{r+1/2}}{2\pi\sqrt{r}(c_4)^{c_4 r+1/2}(1-c_4)^{r-c_4 r+1/2}} \\
&= \frac{e}{2\pi\sqrt{r c_4(1-c_4)}(c_4)^{c_4 r}(1-c_4)^{r-c_4 r}} \\
&= \frac{e^{c_4 r \ln\left(\frac{1}{c_4}\right)+(1-c_4) r \ln\left(\frac{1}{1-c_4}\right)}}{2\pi e^{-1}\sqrt{r c_4(1-c_4)}}
\end{aligned}$$

Thus, by union bounds the probability that there exists $X \subseteq I^*_v(r)$ and $Y \subseteq I^*_{v+r}(r)$ s.t. $|X| = |Y| = c_4 r$ and $X \times Y \cap E_m = \emptyset$ is at most

$$\begin{aligned}
\left(\frac{1}{e}\right)^{\frac{\tau(1-\gamma)^2 c_4^2 r}{8}} \binom{r}{c_4 r}^2 &\leq \frac{e^{2c_4 r \ln\left(\frac{1}{c_4}\right)+2(1-c_4) r \ln\left(\frac{1}{1-c_4}\right)-\frac{\tau(1-\gamma)^2 c_4^2 r}{8}}}{\left(2\pi e^{-1}\sqrt{r c_4(1-c_4)}\right)^2} \\
&= \frac{x^r}{4\pi^2 e^{-2} r c_4(1-c_4)} .
\end{aligned}$$

The probability that a node y is not a (c_4, r^*)-local expander is at most

$$\begin{aligned}
\sum_{r=r^*}^{n'} \frac{x^r}{4\pi^2 e^{-2} r c_4(1-c_4)} &= \frac{1}{4\pi^2 e^{-2} c_4(1-c_4)} \sum_{r=r^*}^{n'} x^r/r \\
&\leq \frac{1}{4r^*\pi^2 e^{-2} c_4(1-c_4)} \left(\frac{x^{r^*} - x^{n'+1}}{1-x}\right) \\
&\leq \frac{1}{4r^*\pi^2 e^{-2} c_4(1-c_4)} \left(\frac{x^{r^*}}{1-x}\right) = c_6
\end{aligned}$$

□

Assuming we can find appropriate constants, the Theorem 3.1 now follows directly from the above claims. Lemma 3.3 implies that $(1-c_6)n'$ nodes in G_m are (c_4, r^*)-local expanders in expectation we expect to have at least $n' - |S|\frac{2}{c_5} - c_6 n'$ nodes that are both (c_4, r^*)-local expanders and c_5-good with respect to S. Lemma 3.2 then implies that there is a path running through each of these nodes. Thus, the meta-graph G_m of a random DRSample DAG G is $(e, \mathbf{d})$-depth robust where $\mathbf{d}$ is a random variable with expectation $\mathbf{E}[\mathbf{d}] = n' - |S|\frac{2}{c_5} - c_6 n'$. Furthermore, Lemma C.1 from the Appendix C allows us to claim that for any constant $\epsilon > 0$ we have $\mathbf{d} \geq n' - |S|\frac{2}{c_5} - c_6 n' - \epsilon n$ except with negligible probability in n assuming that $x < 1$. Thus, G_m is $(c_1 n', c_2 n')$-depth robust with $c_2 = 1 - \frac{2c_1}{c_5} - c_6 - \epsilon$. By Claim 1 G is $(c_1 n', c_2 \gamma m n', m)$-block depth robust or simply $\left(\frac{c_1 n}{\tau \log n}, c_2 \gamma n\right)$-depth-robust. This in turn implies that $\mathsf{aAT}(G) \geq c_3 n^2/\log n$ where $c_3 = c_1 c_2 \gamma/\tau$. It remains to find suitable constants $\gamma, \tau, c_1, c_2, c_3, c_4, c_5, c_6$ and r^* to maximize c_3 such that all of the following additional constraints are satisfied:

(1) $0 < c_1, c_2, c_3, c_4, c_5, c_6 < 1, \epsilon > 0$,
(2) $c_2 \leq 1 - \frac{2c_1}{c_5} - c_6 - \epsilon$ from Lemma C.1 ,
(3) $c_3 \leq c_1 c_2 \gamma/\tau$ from Theorem 2.4
(4) $3c_4 + c_5 \leq 1$ from Lemma 3.2
(5) $\left(2c_4 \ln\left(\frac{1}{c_4}\right) + 2(1-c_4)\ln\left(\frac{1}{1-c_4}\right) - \frac{\tau(1-\gamma)^2 c_4^2}{8}\right) < \ln(0.861)$ from Lemma 3.3.
(6) $c_5 r^* \leq 1$ from Lemma 3.2.
(7) $c_6 \geq \frac{(0.861)^{r^*}}{4r^*\pi^2 e^{-2} c_4(1-c_4)(1-0.861)}$ from Lemma 3.3.

The proof now follows by setting $\tau = 160, r^* = 8, c_4 = 0.2916, c_5 = 0.125, \gamma = 0.1, c_6 = 0.00861, \epsilon = 0.01, c_2 = 0.3, c_1 = 0.038945$ and $c_3 = 7.3 \times 10^{-6}$.

4 GADGET TO BOOST CC

This goal of this section is to build a DAG on $n \in \mathbb{N}$ nodes with high exact aAT suitable for use in the construction of memory-hard functions. Theorem 2.4 implies[8] that if a DAG G is $(c_1 n/\log n, c_2 n)$-depth-robust for some constants c_1 and c_2 then $\Pi^{\|}_{cc}(G) \geq \frac{c_1 c_2 n^2}{\log n}$. Combined with the Theorem 3.1 this implies that algorithm DRSample already samples a low indegree, simple and locally navigable graph. Moreover its aAT is bounded by the exact (i.e. non-asymptotic) term $c_1 c_2 e d$. Our next theorem shows that we can do better by modifying (any block-depth-robust) construction to essentially eliminate the dependence on c_2 and directly construct a family of graphs for which $\Pi^{\|}_{cc}(G_n) \geq \frac{c_1(1-\epsilon)n^2}{4\log n} + o(1)$.

The function aATSample (c.f. Algorithm 2) takes as input a graph $H = (\bar{V}, \bar{E}) \in \mathbb{G}_{n,2}$ with node set $\bar{V} = [n]$ and returns a graph $G = (V, E) \in \mathbb{G}_{2n,2}$ with node set $V = [2n]$. It expects access to a function $\mathsf{GetParent}_H(v, i)$ which returns the i^{th} parent of node $v \in \bar{V}$ in H which it uses to implement G's analogous parent function $\mathsf{GetParent}_G(v, i)$ which is parametrized by a constant $c \in (0, 1)$ given as input to aATSample.

Theorem 4.1. *Let $H \in \mathbb{G}_{n/2,2}$ with nodes $\bar{V} = [n/2]$ be*

$$(c_1 n/\log n, c_2 n, c_3 \log n)\text{-block depth-robust}$$

for constants c_1, c_2 and c_3 and let $G = \mathsf{aATSample}(H, c_3)$. Then $G \in \mathbb{G}_{n,2}$ and for every $\epsilon > 0$ if $n > 2^{\left(\frac{1-\epsilon}{c_2 c_3}\right)}$ then $\Pi^{\|}_{cc}(G_n) \geq \frac{c_1(1-\epsilon)n^2}{4\log n}$.

We sketch the proof below and begin with the construction of G. (The pseudocode describing the construction can be found in Algorithm 2.) Let $n' = n/2$. We start with a graph $H_{n'} = (\bar{V}, \bar{E})$ with $\bar{V} = [n']$. We will add n' nodes to form $G = (V, E)$ with $V = [n]$ initially setting $E = \bar{E}$. Setting $n' = n/2$ for each $i \in [n']$ we add the edge $(n' + (i-1), n' + i)$ to E. Next let $m = \lfloor c_3 \log n \rfloor$ and let $M = \{u \in [n] \ : u \equiv 0 \mod m\} = \lfloor n/m \rfloor$. For each $u = bm \in [n']$ and every $v \in [n'+1, n]$ such that $v - n' \equiv b \mod |M|$ we add the edge (u, v) to E.

Clearly, G has $\mathsf{indeg}(G) = 2$. For a node $v \in [n']$ we have that $\mathsf{indeg}(v) \leq \mathsf{indeg}(H_n) \leq 2$. For $v[n'+1, n]$ we have at most 2 incoming edges; $(v-1, v)$ and (bm, v) where $b = v - n' \mod |M|$.

Fix a pebbling $P_1, \ldots, P_t$ of G and let t_i denote the first time step during which node i was pebbled. In the following discussion set $e = c_1 n'/\log n$, $d = c_2 n'$ and $b = c_3 \log n'$. Suppose that for some $i \in [t_{n'}, t_{n-|M|}]$ we have that $|P_i| \leq (1-\epsilon)e$. Then we can show

[8] by setting $S = \emptyset$ and letting T be the sinks of G

Algorithm 2: An algorithm for sampling a high aAT graph.

Function aATSample$(H = (\bar{V} = [n], \bar{E}), c \in (0,1))$**:**
  $V := [2n]$
  $E := \bar{E} \cup \{(i, i+1) : i \in [2n-1]\}$
  for $v \in [n+1, 2n]$ *and* $i \in [2]$ **do** // Populate new edges of graph.
    $E := E \cup \{(v, \text{GetParent}^c(v,i))\}$ // Get i^{th} parent of node v
  end
  return $G := (V, E)$.

Function GetParent$^c(v,i)$**:**
  if $i = 1$ **then**
    $u := i - 1$
  end
  else if $v \le n$ **then**
    $u := \text{GetParent}_H(v, i)$ // DRSample
  end
  else
    $m := \lfloor c \log(n) \rfloor$
    $b := (v - n) \mod \lfloor n/m \rfloor$
    $u := bm$
  end
  return u

that

$$\sum_{j=t_i}^{t_{i+|M|}} \left|P_j\right| \ge \epsilon e d \ .$$

In particular, let $G' = G - V \backslash \text{ancestors}_{G-P_i}([i, i+|M|])$. We note that $H_n - P_i$ is still $(\epsilon e, d, b)$-block-depth-robust. Thus, G' is at least $(\epsilon e, d)$-depth-robust so Theorem 2.4 implies that

$$\sum_{j=t_i}^{t_{i+|M|}} \left|P_j\right| \ge \Pi_{cc}^{\|}(G') \ge \epsilon e d \ .$$

Let $\ell_1 \ge t_{n/2}$ be the smallest time step for which $\left|P_{\ell_1}\right| \le (1-\epsilon)e$ if such a round exists and let v_1 be given such that $t_{v_1} < \ell_1 \le t_{v_1+1}$. In general, once $\ell_1 < \ldots < \ell_i$ and $v_1, \ldots, v_i$ have been defined let ℓ_{i+1} be the least pebbling round such that $\ell_{i+1} \ge t_{v_{i+1}+|M|+1}$ and $\left|P_{\ell_{i+1}}\right| \le (1-\epsilon)e$ if such a round exists. Finally, let v_{i+1} be given such that $t_{v_{i+1}} < \ell_1 \le t_{v_{i+1}+1}$. Continue until we have defined a maximal sequence $\ell_1, \ldots, \ell_{i^*}$.

We have

$$\sum_{j=1}^{t} \left|P_j\right| \ge (1-\epsilon)e \max\{n/2 - mi^*, 0\} + i^* \times \epsilon e d \ge \min\left\{\frac{e(1-\epsilon)n}{2}, \frac{n}{2|M|} ed\right\}.$$

We have $\frac{n}{2|M|} ed = \Omega\left(n^2\right)$ and $\frac{e(1-\epsilon)n}{2} = O\left(\frac{n}{\log n}\right)$. Thus, we can find $N > 0$ such that for all $n > N$ the second term dominates and we have

$$\sum_{j=1}^{t} \left|P_j\right| \ge \min\left\{\frac{e(1-\epsilon)n}{2}, \frac{n}{2|M|} ed\right\} \ge \frac{c_1(1-\epsilon)n^2}{4 \log n} \ .$$

In particular this holds for any $n > 2^{\left(\frac{1-\epsilon}{c_2 c_3}\right)}$.

REMARK 1. *We note that* aATSample *(c.f. Algorithm 2) yields a graph G which, except with negligible probability in n, has* $\text{aAT}(G) \ge 6.08 \times 10^{-5} n^2 / \log n$. *In terms of provable security guarantees the lower bound for* aATSample *improves on our lower bound for* DRSample *by an order of magnitude, and it appears to improve on [27] by two orders of magnitude — see Appendix D. Interestingly, while we can establish better constants in our proof of security for* aATSample *the simpler construction* DRSample *appears to perform better in practice.*

5 EMPIRICAL ANALYSIS

In this section we describe the experiments we ran comparing various graphs and interpret their results.

Depth-Reduction and aAT Attacks. To empirically investigate the depth-robustness of graphs we implemented five algorithms for constructing a depth-reducing set S for a DAG G. Each attack takes as input a target depth d_{tgt} and outputs a set S such that $\text{depth}(G - S) \le d_{tgt}$ The first is the layered attack of Alwen and Blocki [5], which has been shown to have good asymptotic performance against Argon2i-A [5] and Argon2i-B [6].

The next three attacks are based on Lemma 2.2 (Valiant's Lemma Attacks), which has been used [5] to provide a generic upper bound on the aAT of *any* DAG G with constant indegree: $\text{aAT}(G) = O\left(\frac{n^2 \log\log n}{\log n}\right)$. The first variant (Lazy Valiant) simply sets $b = 2$, computes sets $S_1, \ldots, S_{\lceil \log_b(\text{depth}(G)) \rceil}$, sets $T_0 = \emptyset$ and greedily updates $T_{i+1} := T_i \cup \text{argmin}_{j \notin T_i} \left|S_j\right|$ until $\text{depth}(G - \bigcup_{j \in T_i} S_j) \le d_{tgt}$. The second variant is the same except that we set $b = 3$. Finally, the third variant is the one described by Alwen and Blocki [5]. Briefly, if we let $G_0 = G$ then we can apply one round of Valiant's Lemma ($b = 2$) to obtain a set S_0 such that $\text{depth}(G_0 - S_0) \le 2^{\lceil \log_b(\text{depth}(G)) - 1 \rceil}$. Setting $G_{i+1} = G_i - S_i$ we can iterate until we obtain a graph $G_k = G - \bigcup_{i=0}^{k-1} S_i$ with $\text{depth}(G_k) \le d_{tgt}$. While the theoretical behavior of Valiant's Lemma against highly depth-robust DAGs is well understood in an asymptotic sense, to the best of our knowledge our experiments give the first empirical results about the behaviour of these attacks in practice.

The fifth attack is a hybrid which combines the layered attack and Valiant's Lemma Attack. Briefly, the attack partitions the nodes of the graph into $\sqrt{d_{tgt}}$ layers as in [5] and then uses Lemma 2.2 to reduce the depth of each layer to $\sqrt{d_{tgt}}$ to that the resulting graph has depth at most d_{tgt}. Finally, the sixth "Best Attack" algorithm we implemented simply takes the best (smallest) S produced by any of the four previous algorithms.

To empirically investigate the aAT of our candidate graphs we implemented AB16 pebbling algorithm of [5] which has been shown to be quite effective against many practical iMHFs both theoretically [5, 7, 9] and empirically [6]. This algorithm takes as input a depth reducing set S as well as a key parameter $g \in [\text{depth}(G - S), n]$, which specifies the length of each "light phase" in the attack, and outputs a pebbling P with cost $\text{aAT}(P) = O\left(|S|n + gn + \frac{n^2 d}{g}\right)$. Our implementation takes as input a DAG G on $n = 2^k$ nodes with $k \in [14, 24]$. In our attack we enumerate over target depths $d_{tgt} \in \{2^i : i \in [3, k-2]\}$ and for each target depth we use the "Best Attack" heuristic to construct the smallest depth-reducing

set S. Next we iterate over values of the remaining key parameter $g \in [\text{depth}(G-S), n]$ and run the Alwen-Blocki attack [5] — our implementation includes the heuristic optimization from [6]. Finally, we output the pebbling P with minimal aAT(P) over all choices of the key parameters d_{tgt} and g.

Candidate Graphs. As test subjects we implemented a total of 8 graphs. As a benchmark we implemented the DAGs underlying Argon2i-A and Argon2i-B. We also implemented the DAGs sampled by DRSample and aATSample.

In certain practical situations where trusted uniform randomness is at a premium it may be beneficial to have graphs which require less randomness to sample. To that end we also implemented hybrid versions HDRSample and HaATSample of our two main constructions. In a nutshell the hybrid versions differ from the originals in that they now select the range sizes for long edges deterministically rather than randomly. Their full pseudo-code can be found in the appendix in Algorithm 3 and Algorithm 5, respectively. The formal results in the previous sections for DRSample and aATSample carry over almost unchanged to their hybrid counterparts. Recall that the algorithm aATSample takes as input a parameter $c \in (0, 1)$, we evaluate the algorithm using the parameters $c \in \{0.1, 0.5\}$.

Finally, for applications where randomness is simply not available, we have also implemented an (exceedingly simple and efficient) fully deterministic graph Deterministic. In this graph the second incoming edge of each consecutive node has double the length of the second edge for the previous node. This length doubling is repeated until edges no longer fit in the graph at which point the next edge has length 2 again and the doubling process begins anew. Intuitively this approximates the distribution of the edge lengths of DRSample as that can be seen as first uniformly samples a power of two to set a range size and then uniformly samples an edge length within that range size.

Results. We describe three figures which summarizing the results of our experiments.

Figure 1 compares the depth-reducibility of each of the candidate DAGs. In particular, we plot depth vs the minimum size of a depth reducing set found by any of the five attacks. We see that all new constructions and their variants have better resistance to the depth-reduction attacks than both Argon2i variants. In particular, DRSample and HDRSample provide the strongest resistance.

Figure 2 plots the quality of the best attack we found. (Recall that quality compares the attacker's aAT against the honest algorithm's aAT. A parallel pebbling attack with 3 means it is 3 times more efficient than the strategy used by honest (sequential) parties when pebbling the graph.). Once again we see that all new constructions and their variants have higher aAT cost (smaller attack quality) than Argon2i-A and Argon2i-B meaning that they are more resistant to attacks. Once again DRSample and HDRSample offer the strongest resistance.

Figure 3 compares the empirical performance of the layered depth-reducing attack [5] with the performance of the Lazy Valiant attack. In particular, we plot depth vs the minimum size of a depth reducing set found by the two attacks for 4 of the candidate graphs. As the plot shows, Valiant's attack consistently outperforms the layered attack of Alwen and Blocki [5]. Valiant's lemma is dramatically superior when we are attacking highly depth-robust constructions

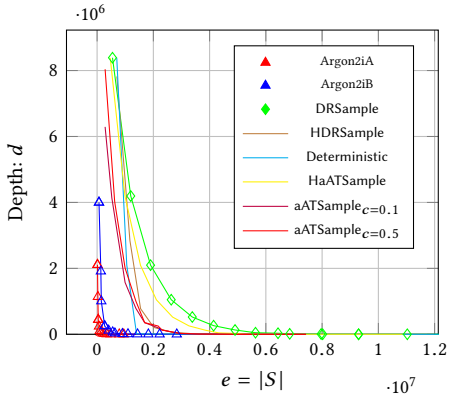

Figure 1: Depth-Reducibility of DAGs under Best Attack ($n = 2^{24} \approx 10^{7.224}$)

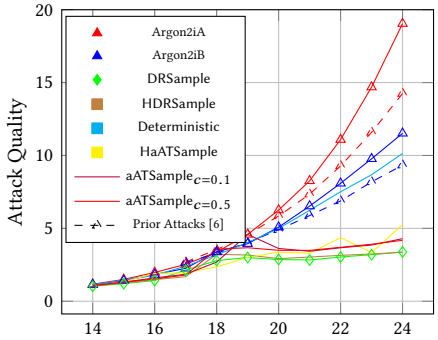

Figure 2: Attack Quality vs iMHF Candidates

such as DRSample or Deterministic. We used diamonds to highlight the curves for DRSample and Deterministic under the layered attack, since these curves are so close to the bottom right side of the plot that they are difficult to see. Surprisingly, our results show that in practice Valiant's attack also appears to perform *slightly* better than the Layered attack against Argon2i-A and Argon2i-B. This, despite the (known) theoretical upper bounds on $|S|$ for those DAGs being much smaller for the Layered attack. Based on these empirical results we conjecture that one could provide significantly tighter theoretical upper bounds on the size of the set S we obtain when we use Lazy Valiant to build depth-reducing sets for Argon2i-A and B. The only known theoretical guarantee is that Valiant's lemma yields a set $|S| \leq O\left(\frac{n \log d_{tgt}}{\log n}\right)$ such that $\text{depth}(G-S) \leq d_{tgt}$.

6 IMPLEMENTATION

To implement our iMHF we modified Argon2i [14] replacing its edge structure with that of DRSample. We selected DRSample because it is simple and our empirical analysis suggests that it offers the best resistance to attacks. Our source code is available at [**?**]. The modification involved adding about 5 lines of code for data independent addressing, and commenting out over thirty lines of code.

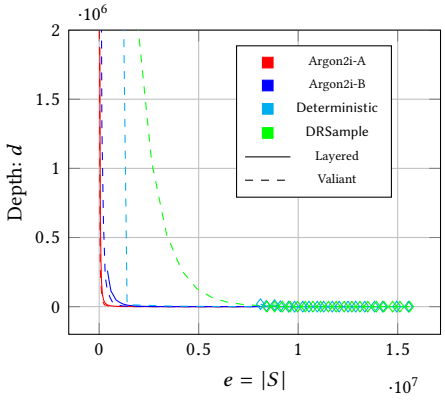

Figure 3: Depth-Reducibility: Layered vs. Lazy Valiant Depth-Reduction Attacks ($n = 2^{24} \approx 10^{7.224}$)

Remark 1: Part of the reason for the dramatic reduction in lines of code is that we decided to remove support for multiple lanes (e.g., to support parallelism) as researchers have previously raised concerns [6] about the high depth-reducibility of an Argon2i DAG when the number of lanes becomes large[9]. At this time we would instead recommend supporting parallelism using the trivial approach: instantiate several independent iMHF threads (with different salt values) and hash each of these outputs to produce the final output block.

Remark 2: While our primary goal was to implement our iMHF DRSample, the reference code still includes the data-dependent modes of operation. However, the data-dependent mode of operation uses the uniform distribution over edges similar to SCRYPT. The recent security proof for SCRYPT [8] suggests that a uniform edge distribution is the right distribution to use for data-dependent modes. While SCRYPT has optimal aAT complexity, it is liable to space-time tradeoff attacks (e.g., an attacker could compute the function in n^2 steps with maximum memory usage $O(1)$). We conjecture that the "id" mode of operation, which runs data-independent mode for $n/2$ steps before switching to the data-dependent mode of operation, might provide much stronger resistance to space-time trade-off attacks though we leave this question as an interesting direction for future research.

6.1 Timing Results.

Recall that the goal when designing an iMHF is to find a function which, for a fixed amount of time used for honest sequential evaluation (e.g. 1/2 second) forces the maximum cost possible per rate of evaluation on an attacker. Thus we also compared the running time of the honest evaluation of an iMHF using our new DAGs with Argon2i-B. In a nutshell we found the new DAGs to actually be *slightly* but noticeably faster than Argon2i-B. This despite, our empirical (and previous theoretic) evidence indicating that for any given set of parameters the new iMHFs incur significantly greater amortized area-time complexity for the adversary.

In more detail we modified the Argon2i-B implementation [14] replacing its edge structure with that of DRSample. We compared the time required on an Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 8GB of memory to evaluate the two iMHFs for a setting with a single lane, 1GB of memory (i.e. 2^{20} blocks). We found that the performance of the new iMHF is actually *marginally* better than that of Argon2i-B (0.969 seconds for original vs. 0.966 seconds for modified version). (While the difference is small we do get statistically significant evidence for the hypothesis that the new iMHF is even slightly faster.)

7 OPEN QUESTIONS

While there is still a large gap between the constant factors in the best known *provable* lower bounds on aAT for DRSample and aATSample and the best known upper bounds (attacks) on aAT, we are conjecturing that both constructions aATSample and DRSample can provide strong memory hardness guarantees in practice.

How close to optimal are the depth-reducing attacks on DAGs like DRSample and Deterministic? We conjecture that the constructions of depth-reducing sets are *nearly* optimal. If this conjecture is true it would imply that for $n \leq 2^{24}$ the quality of any attack on DRSample is at most 28 [7]. Thus, an important open challenge is to find smaller depth-reducing set for the specific graph DRSample with $n = 2^{24}$ nodes (or demonstrate that no smaller depth-reducing set exists). Is there an efficient approximation algorithm to find a small depth-reducing set S given a target depth d_{tgt}? While it is hard to approximate $|S|$ to within a factor of 1.99 [17], it is still possible that an efficient 2.01-approximation algorithm exists. Similarly, is there an efficient algorithm to approximate the aAT of a fixed DAG G? Blocki and Zhou [17] recently showed that aAT is NP-hard to compute exactly, but an efficient approximation algorithm would allow us to quickly analyze constructions like DRSample.

Another interesting theoretical challenge is to construct a family of $\left(\frac{4n}{\log n}, \Omega\left(\frac{n}{\log^{1-\epsilon} n}\right), \log n\right)$-block depth robust DAGs. In fact, we conjecture that DRSample already satisfies this property. Such a family could be used along with aATSample to obtain a family of DAGs with aAT *provably* at least $\frac{(1-\epsilon)n^2}{\log n}$. [10]

Finally, we conjecture that our constructions of practical depth-robust DAGs might lead to the development of data-dependent MHFs with *provably* strong resistance to space-time trade-off attacks.

[9] In particular, any directed edge from a node in lane i to a node in lane $j \neq i$ cannot exist in the same slice. When sampling a backedge for a node v in lane j Argon2 follows the following approach: (1) select a random lane ℓ, (2) select a (random) predecessor $r(v, j) \in [v-2]$ of v in lane ℓ and add the directed edge $((r(v, j), \ell), (v, j))$ from node $(r(v, j), \ell)$ to (v, j). If $\ell \neq j$ then it is required that $v - r(v, j)$ is large to avoid deadlocking each thread (e.g., by ensuring that the value for node $(r(v, j), \ell)$ is certainly computed in thread ℓ before we try to compute the value for node (v, j) in thread j. Alwen and Blocki [6] observed that as the number of threads grows large *almost* all backedges are long, which makes it significantly easier to construct depth-reducing sets for their attack.

[10] In particular, it would cost $\Omega\left(\frac{n^2}{\log^{2-\epsilon} n}\right)$ to re-pebble the DAG starting with $\frac{4(1-\epsilon)n}{\log n}$ pebbles on the graph. The graph aATSample constructs would consist of a gadget that forces us to either keep $\frac{2(1-\epsilon)n}{\log n}$ pebbles on the DAG during the last $n/2$ steps or re-pebble within the next $\frac{n}{2\log n}$ steps. It would cost $\Omega\left(\frac{n^2}{\log^{1-\epsilon} n}\right)$ to re-pebble every $\frac{n}{\log n}$ steps so it is better to keep $\frac{2(1-\epsilon)n}{\log n}$ pebbles around for these last $n/2$ steps. Thus, aAT $\geq \frac{(1-\epsilon)n^2}{\log n}$.

A MEMORY-HARD VS. MEMORY-BOUND

We show that memory-bound and memory-hard are distinct complexity notions by giving a simple and intuitive separating example; that is a function which is memory-bound but not memory-hard.

Recall that a function f_n with hardness parameter $n \in \mathbb{N}$ is memory-bound if the expected number of cache misses required to compute f_n on fresh input (in the random oracle model) grows linearly in n. Conversely f_n is memory-hard if, roughly speaking, the product of (parallel) space-time grows roughly quadratically in n.

Consider the function f_n given by a array A of s uniform random w-bit values. The function f_n on input x with random oracle H is defined as follows. Let $b_0 = x$. For $i \in [n]$ let $b_i = a_j$ where $j = H(b_{i-1}) \mod s$ and set $f_n(x) = b_n$. Roughly speaking this is the memory-bound function given in [23] and the authors show that if $w * s$ is at least twice the size of cache then the expected number of cache misses grows linearly in n. In other words f_n is memory-bound.

However obvious (sequential) algorithm which computes f_n by computing the b_i values in increasing order of their index shows that f_n is *not* memory-hard. Indeed, the time complexity of this algorithm grows linearly in n but its space complexity remains constant in n. Thus the product of its space-time grows only linearly in n.

B EXTRA GRAPH CONSTRUCTIONS

Algorithm 3: An alternative algorithm for sampling depth-robust graphs.

Function HDRSample($n \in \{2^i : i \in \mathbb{N}_{\geq 1}\}$):
  $V := [v]$
  $E := \{(1,2)\}$
  for $v \in [3,n]$ *and* $i \in [2]$ **do** `// Populate edges of graph.`
    $E := E \cup \{(v, \text{GetParent}(v,i))\}$ `// Get` i^{th} `parent of node` v
  end
  return $G := (V, E)$.

Function GetParent(v,i):
  if $i = 1$ **then**
    $u := i - 1$
  else
    $g' := v \mod \log_2(n)$ `// Select range size.`
    $g := \min(v, 2^{g'+1})$ `// Don't make edges too long.`
    $r \leftarrow [g/2, g]$ `// Select random edge length.`
  end
  return $v - r$

C CONCENTRATION BOUNDS

Lemma C.1. *Let $G \leftarrow \text{DRSample}(n)$ and let $m = (\tau+1)\log n$, $r^* = O(1)$ and let $x \in [n'] = [n/m]$ be a meta-node then x is a (c_4, r^*)-local expander in the meta-graph $G_m = (V_m, E_m)$ with probability at most*

Algorithm 4: A deterministic algorithm for sampling (conjectured) depth-robust graphs.

Function Deterministic($n \in \{2^i : i \in \mathbb{N}_{\geq 1}\}$):
  $V := [v]$
  $E := \{(1,2)\}$
  for $v \in [3,n]$ *and* $i \in [2]$ **do** `// Populate edges of graph.`
    $E := E \cup \{(v, \text{GetParent}(v,i))\}$ `// Get` i^{th} `parent of node` v
  end
  return $G := (V, E)$.

Function GetParent(v,i):
  if $i = 1$ **then**
    $u := i - 1$
  else
    $j := v \mod \log_2(n)$
  end
  return $v - \min\{2^j, v-1\}$

$c_6 = \frac{1}{4r^*\pi^2 e^{-2} c_4(1-c_4)} \left(\frac{x^{r^*}}{1-x}\right)$ *where* $x = e^{\left(2c_4 \ln\left(\frac{1}{c_4}\right) + 2(1-c_4)\ln\left(\frac{1}{1-c_4}\right) - \frac{\tau(1-\gamma)^2 c_4^2}{8}\right)}$. *Furthermore, for any $\epsilon > 0$ we have that, except with negligible probability in n, at least $n'(1-c_6-\epsilon)$ nodes in the meta-graph are (c_4, r^*)-local expanders.*

Proof. (sketch) We first show that except with negligible probability every node is a $(c_4, n^{1/4})$-local expander.

Claim 3. *Except with probability*

$$\frac{n^{3/4}}{4\pi^2 e^{-2} c_4(1-c_4)} \left(\frac{x^{n^{1/4}}}{1-x}\right)$$

every node $v \in V_m$ is a $(c_4, n^{1/4})$-local expander.

The proof of Claim 3 closely follows the proof of Lemma 3.3. It is included below for completeness. Let EXPAND be the event that all metanodes are $(c_4, n^{1/4})$-local expanders and let EXP_{u,r^*} be the indicator random variable for the event that node u is a(c_4, r^*)-local expander. Conditioning on the event EXPAND the events EXP_{u,r^*} and EXP_{v,r^*} are independent whenever $|v-u| \geq 4n^{1/4}$. We can now set $B_j = \sum_i \text{EXP}_{j+4in^{1/4},r^*}$ for each $j \leq 4n^{1/4}$. Since B_j is the sum of independent random variables we can apply chernoff bounds+ union bounds to show that except with negligible probability we have $\mathbf{E}\left[B_j\right] - B_j \leq \epsilon \frac{n'}{4n^{1/4}}$ for each $j \leq 4n^{1/4}$. It follows that $\sum_j B_j \geq (1-c_6-\epsilon)n'$ except with negligible probability in n. □

Proof of Claim 3. Fix a node $v \in V_m$ and $r \geq n^{1/4}$. Let i be given such that $2^{i+1} \geq 2rm \geq 2^i$. Fix $X \subseteq I^*_v(r)$ and $Y \subseteq I^*_{v+r}(r)$ then we have

$$\Pr\left[X \times Y \cap E_m = \emptyset\right] \leq \left(1 - \frac{|X|(1-\gamma)}{8r\log n}\right)^{(1-\gamma)m|Y|} \leq \left(\frac{1}{e}\right)^{\frac{(1-\gamma)^2|Y||X|\tau}{8r}}$$

Algorithm 5: A alternative algorithm for sampling a high aAT graph.

Function HaATSample($H = (\bar{V} = [n], \bar{E})$, $c \in (0,1)$):
  $V := [2n]$
  $E := \bar{E} \cup \{(i, i+1) : i \in [2n-1]\}$
  for $v \in [n+1, 2n]$ *and* $i \in [2]$ **do** // Populate new edges of graph.
    $E := E \cup \{(v, \text{GetParent}^c(v,i))\}$ // Get i^{th} parent of node v
  end
  return $G := (V, E)$.

Function GetParent^c(v,i):
  if $i = 1$ **then**
    $u := i - 1$
  end
  else if $v \leq n$ **then**
    $g' \leftarrow [1, \lfloor \log_2(v) \rfloor + 1]$ // Select random range size.
    $g := \min(v, 2^{g'})$ // Don't make edges too long.
    $r \leftarrow [\max(g/2, 2), g]$ // Select random edge length.
    $u := v - r$
  end
  else
    if $v = 1 \mod 2$ **then**
      $u \leftarrow [n]$
    end
    else
      $g' \leftarrow [1, \lfloor \log_2(v) \rfloor + 1]$ // Select random range size.
      $g := \min(v, 2^{g'})$ // Don't make edges too long.
      $r \leftarrow [\max(g/2, 2), g]$ // Select random edge length.
      $u := v - r$
    end
  end
  return u

If we set $|X| = |Y| = c_4 r$ then we have

$$\Pr[X \times Y \cap E_m = \emptyset] \leq \left(\frac{1}{e}\right)^{\frac{(1-\gamma)^2 c_4^2 r \tau}{8}}.$$

We would like to use union bounds to show that (whp) no such sets X, Y exist. We have $\binom{r}{c_4 r}^2$ such pairs X, Y where, by Sterling's inequalities $\sqrt{2\pi} n^{n+0.5} e^{-n} \leq n! \leq e n^{n+0.5} e^{-n}$, we have

$$\begin{aligned}
\binom{r}{c_4 r} &= \frac{r}{(c_4 r)!(r - c_4 r)!} \\
&\leq \frac{e r^{r+1/2}}{\sqrt{2\pi}(c_4 r)^{c_4 r + 0.5}\sqrt{2\pi}(r - c_4 r)^{r - c_4 r + 0.5}} \\
&= \frac{e r^{r+1/2}}{2\pi\sqrt{r}(c_4)^{c_4 r + 1/2}(1-c_4)^{r - c_4 r + 1/2}} \\
&= \frac{e}{2\pi\sqrt{r c_4 (1-c_4)}(c_4)^{c_4 r}(1-c_4)^{r - c_4 r}} \\
&= \frac{e^{c_4 r \ln\left(\frac{1}{c_4}\right) + (1-c_4) r \ln\left(\frac{1}{1-c_4}\right)}}{2\pi e^{-1}\sqrt{r c_4 (1-c_4)}}
\end{aligned}$$

Thus, by union bounds the probability that there exists $X \subseteq I^*_v(r)$ and $Y \subseteq I^*_{v+r}(r)$ s.t. $|X| = |Y| = c_4 r$ and $X \times Y \cap E_m = \emptyset$ is at most

$$\begin{aligned}
\left(\frac{1}{e}\right)^{\frac{\tau(1-\gamma)^2 c_4^2 r}{8}} \binom{r}{c_4 r}^2 &\leq \frac{e^{2c_4 r \ln\left(\frac{1}{c_4}\right) + 2(1-c_4) r \ln\left(\frac{1}{1-c_4}\right) - \frac{\tau(1-\gamma)^2 c_4^2 r}{8}}}{\left(2\pi e^{-1}\sqrt{r c_4 (1-c_4)}\right)^2} \\
&= \frac{x^r}{4\pi^2 e^{-2} r c_4 (1-c_4)}.
\end{aligned}$$

The probability that a there exists a node y that is not a $(c_4, n^{1/4})$-local expander is at most

$$\begin{aligned}
n \sum_{r=n^{1/4}}^{n'} \frac{x^r}{4\pi^2 e^{-2} r c_4 (1-c_4)} &= \frac{n}{4\pi^2 e^{-2} c_4 (1-c_4)} \sum_{r=n^{1/4}}^{n'} x^r / r \\
&\leq \frac{n^{3/4}}{4\pi^2 e^{-2} c_4 (1-c_4)} \left(\frac{x^{n^{1/4}} - x^{n'+1}}{1-x}\right) \\
&\leq \frac{n^{3/4}}{4\pi^2 e^{-2} c_4 (1-c_4)} \left(\frac{x^{n^{1/4}}}{1-x}\right),
\end{aligned}$$

where this last term is negligible as long as $x < 1$.

D CONSTANTS FROM OTHER CONSTRUCTIONS

Peeking under the hood of the [27] construction we see that the DAG G' on n' nodes has

$$\text{indeg}(G') = \delta \geq 10\left(\frac{2}{\epsilon_1 \log_2\left(\frac{1}{(1-\epsilon_1)^2}\right)}\right)\log n' + 4 \log n'$$

and is $(\epsilon n', \epsilon n')$-depth robust. After applying indegree reduction [7] we have a DAG G on $n = 2n'\delta$ nodes that is $(\epsilon n', \epsilon n/2)$-depth robust, and hence $\Pi^{\|}_{cc}(G) \geq \epsilon^2 n^2/(4\delta)$. The parameters ϵ and ϵ_1 must be selected subject to the following constraints

- $1 - \epsilon_4 > 2\epsilon$,
- $\epsilon_5 \geq \epsilon_2$,
- $\epsilon_4 > (2\epsilon/\epsilon_3)$,
- $1 - \epsilon_2 - 3\epsilon_3 > \epsilon_5$ and
- $\epsilon_2/5 > \epsilon_1 > 0$

To obtain the best lower bounds we want to maximize

$$c = \frac{\epsilon^2}{10\left(\frac{2}{\epsilon_1 \log_2\left(\frac{1}{(1-\epsilon_1)^2}\right)}\right) + 4}$$

subject to these constraints. The best parameters we were able to find had $c \leq 4.72 \times 10^{-7}$ ($\epsilon = 0.0714$, $\epsilon_1 = 0.05$, $\epsilon_2 = 1/4 - \epsilon$, $\epsilon_3 = 1/6$, $\epsilon_4 = 0.857$ and $\epsilon_5 = 1/4$).

By comparison, the constants in the lower bound on the aAT for Argon2i-A shown in [7] are larger. In particular, it is shown that any legal pebbling must pay

$$\Pi_{cc}^{\|}(G) \geq \frac{n^{5/3}}{9.6 \times 10^7 \log^2 n}$$

REFERENCES

[1] Password Hashing Competition. (????). https://password-hashing.net/.

[2] 2016. Cumulative Space in Black-White Pebbling and Resolution. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, 9-11 January 2017, Berkeley, California USA*.

[3] Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. 2005. Moderately Hard, Memory-bound Functions. *ACM Trans. Internet Technol.* 5, 2 (May 2005), 299–327. https://doi.org/10.1145/1064340.1064341

[4] Leonardo C Almeida, Ewerton R Andrade, Paulo SLM Barreto, and Marcos A Simplicio Jr. 2014. Lyra: Password-based key derivation with tunable memory and processing costs. *Journal of Cryptographic Engineering* 4, 2 (2014), 75–89.

[5] Joël Alwen and Jeremiah Blocki. 2016. Efficiently Computing Data-Independent Memory-Hard Functions. In *Advances in Cryptology CRYPTO'16*. Springer, 241–271.

[6] Joël Alwen and Jeremiah Blocki. 2017. Towards Practical Attacks on Argon2i and Balloon Hashing. In *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P 2017)*. IEEE, (to appear). http://eprint.iacr.org/2016/759.

[7] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. 2017. Depth-Robust Graphs and Their Cumulative Memory Complexity. In *EUROCRYPT (LNCS)*. https://eprint.iacr.org/2016/875.

[8] Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. 2017. `scrypt` is Maximally Memory-Hard. In *Advances in Cryptology-EUROCRYPT 2017*. Springer, (to appear). http://eprint.iacr.org/2016/989.

[9] Joël Alwen, Peter Gaži, Chethan Kamath, Karen Klein, Georg Osang, Krzysztof Pietrzak, Leonid Reyzin, Michal Rolínek, and Michal Rybár. 2016. On the Memory-Hardness of Data-Independent Password-Hashing Functions. Cryptology ePrint Archive, Report 2016/783. (2016). http://eprint.iacr.org/2016/783.

[10] Joël Alwen and Vladimir Serbinenko. 2015. High Parallel Complexity Graphs and Memory-Hard Functions. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing (STOC '15)*. http://eprint.iacr.org/2014/238.

[11] Joël Alwen and Björn Tackmann. 2017. Moderately Hard Functions: Definition, Instantiations, and Applications. (2017). https://eprint.iacr.org/2017/.

[12] Daniel J. Bernstein. Cache-Timing Attacks on AES. (????). http://cr.yp.to/antiforgery/cachetiming-20050414.pdf

[13] Billy Markus. 2013. Dogecoin. (2013). http://dogecoin.com/

[14] Alex Biryukov, Daniel Dinu, Jean-Philippe Aumasson, and Samuel Neves. 2017. Argon2. https://github.com/P-H-C/phc-winner-argon2. (2017).

[15] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. 2016. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. IEEE, 292–302. https://doi.org/10.1109/EuroSP.2016.31

[16] Alex Biryukov and Dmitry Khovratovich. 2015. Tradeoff Cryptanalysis of Memory-Hard Functions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II (Lecture Notes in Computer Science)*, Tetsu Iwata and Jung Hee Cheon (Eds.), Vol. 9453. Springer, 633–657. https://doi.org/10.1007/978-3-662-48800-3_26

[17] Jeremiah Blocki and Samson Zhou. 2016. On the Computational Complexity of Minimal Cumulative Cost Graph Pebbling. *arXiv preprint arXiv:1609.04449* (2016).

[18] Dan Boneh, Henry Corrigan-Gibbs, and Stuart Schechter. 2016. Balloon Hashing: Provably Space-Hard Hash Functions with Data-Independent Access Patterns. Cryptology ePrint Archive, Report 2016/027, Version: 20160601:225540. (2016). http://eprint.iacr.org/.

[19] Donghoon Chang, Arpan Jati, Sweta Mishra, and Somitra Kumar Sanadhya. 2014. Rig: A simple, secure and flexible design for Password Hashing Version 2.0. (2014).

[20] Charles Lee. 2011. Litecoin. (2011). https://litecoin.info/

[21] Stephen A. Cook. 1973. An Observation on Time-storage Trade off. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing (STOC '73)*. ACM, New York, NY, USA, 29–33. https://doi.org/10.1145/800125.804032

[22] Bill Cox. 2014. Twocats (and skinnycat): A compute time and sequential memory hard password hashing scheme. *Password Hashing Competition. v0 edn.* (2014).

[23] Cynthia Dwork, Andrew Goldberg, and Moni Naor. 2003. On Memory-Bound Functions for Fighting Spam. In *Advances in Cryptology - CRYPTO 2003 (Lecture Notes in Computer Science)*, Vol. 2729. Springer, 426–444. https://doi.org/10.1007/978-3-540-45146-4_25

[24] Cynthia Dwork, Moni Naor, and Hoeteck Wee. 2005. Pebbling and Proofs of Work. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings (Lecture Notes in Computer Science)*, Vol. 3621. Springer, 37–54. https://doi.org/10.1007/11535218_3

[25] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. 2015. Proofs of Space. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II (Lecture Notes in Computer Science)*, Rosario Gennaro and Matthew Robshaw (Eds.), Vol. 9216. Springer, 585–605. https://doi.org/10.1007/978-3-662-48000-7_29

[26] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. 2011. One-Time Computable Self-erasing Functions. In *TCC (Lecture Notes in Computer Science)*, Yuval Ishai (Ed.), Vol. 6597. Springer, 125–143.

[27] Paul Erdoes, Ronald L. Graham, and Endre Szemeredi. 1975. *On Sparse Graphs with Dense Long Paths*. Technical Report. Stanford, CA, USA.

[28] Christian Forler, Stefan Lucks, and Jakob Wenzel. 2013. Catena: A Memory-Consuming Password Scrambler. *IACR Cryptology ePrint Archive* 2013 (2013), 525.

[29] Carl E. Hewitt and Michael S. Paterson. 1970. Record of the Project MAC Conference on Concurrent Systems and Parallel Computation. ACM, New York, NY, USA, Chapter Comparative Schematology, 119–127. https://doi.org/10.1145/1344551.1344563

[30] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. 2013. Publicly verifiable proofs of sequential work. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, Robert D. Kleinberg (Ed.). ACM, 373–388. https://doi.org/10.1145/2422436.2422479

[31] Wolfgang J. Paul and Rüdiger Reischuk. 1980. On Alternation II. A Graph Theoretic Approach to Determinism Versus Nondeterminism. *Acta Inf.* 14 (1980), 391–403. https://doi.org/10.1007/BF00286494

[32] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. 1976. Space Bounds for a Game on Graphs. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing (STOC '76)*. ACM, New York, NY, USA, 149–160. https://doi.org/10.1145/800113.803643

[33] C. Percival. 2009. Stronger key derivation via sequential memory-hard functions. In *BSDCan 2009*.

[34] Krisztián Pintér. 2014. Gambit – A sponge based, memory hard key derivation function. Submission to Password Hashing Competition (PHC). (2014).

[35] Georg Schnitger. 1982. A Family of Graphs with Expensive Depth Reduction. *Theor. Comput. Sci.* 18 (1982), 89–93. https://doi.org/10.1016/0304-3975(82)90113-X

[36] Georg Schnitger. 1983. On Depth-Reduction and Grates. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 323–328. https://doi.org/10.1109/SFCS.1983.38

[37] Leslie G. Valiant. 1977. Graph-Theoretic Arguments in Low-Level Complexity. In *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings (Lecture Notes in Computer Science)*, Jozef Gruska (Ed.), Vol. 53. Springer, 162–176. https://doi.org/10.1007/3-540-08353-7_135

[38] Vitalik Buterin. 2013. Ethereum. (2013). https://www.ethereum.org/

[39] Hongjun Wu. 2015. POMELO – A Password Hashing Algorithm. (2015).

[40] Zerocoin Electric Coin Company. 2016. ZCash. (2016). https://z.cash/